**Objective:**

The objective of this project is to design and develop a software-only solution for a smart college campus that integrates a simulated surveillance system for monitoring and an interactive guidance system to assist students, staff, and visitors in navigating the campus efficiently.

**Abstract:**

The **Surveillance and Guidance System for Smart College Campus** aims to enhance campus safety, accessibility, and navigation by utilizing cutting-edge technologies. The system simulates real-time surveillance using pre-recorded video feeds for monitoring activities and provides an interactive digital map for navigation. This software-based approach avoids hardware dependencies, making it an affordable and scalable solution for modern educational institutions. Features like event notifications, interactive campus maps, and route guidance ensure a seamless and smart campus experience.

**Features:**

1. **Simulated Surveillance:**
   * Displays video streams mimicking real-time surveillance.
   * Processes feeds to simulate object detection (e.g., highlighting activities).
   * Logs detected events and triggers simulated alerts for unauthorized activity.
2. **Interactive Guidance System:**
   * Provides a digital campus map with key locations (e.g., classrooms, library, cafeteria).
   * Allows users to search for locations and get the shortest path navigation.
   * Implements navigation using Mapbox or Google Maps APIs.
3. **Unified Dashboard:**
   * Combines surveillance feeds and navigation into a single user-friendly interface.
   * Accessible via web browsers for easy use.
4. **Event Logging:**
   * Simulates logging events such as unusual activities detected by the surveillance system.
   * Displays notifications and logs in real time on the dashboard.

**Technology Stack:**

* **Frontend:** React.js, HTML, CSS, JavaScript, Bootstrap.
* **Backend:** Node.js (Express.js).
* **Database:** Firebase Realtime Database or SQLite for event logs.
* **APIs and Tools:**
  + OpenCV.js for video simulation and object detection.
  + Mapbox or Google Maps API for navigation and map rendering.
* **Development Environment:** Visual Studio Code.

**Implementation Details:**

1. **Simulated Surveillance Module:**
   * Uses OpenCV.js to process pre-recorded videos to demonstrate real-time monitoring.
   * Simulates object detection and event triggering for monitoring.
2. **Guidance System Module:**
   * Integrates an interactive map to show important campus locations.
   * Allows users to search for and navigate to specific destinations.
3. **Integration:**
   * A unified dashboard combines the video feed and navigation map.
   * Backend APIs handle communication between the modules.

**Benefits:**

* Improves campus safety by simulating a robust surveillance system.
* Enhances user experience with intuitive campus navigation.
* Provides a scalable and cost-effective solution using software-only resources.

**Applications:**

* Smart college campuses to enhance student and staff experience.
* Virtual campuses or digital tours for prospective students.
* Simulated environments for learning about surveillance and navigation technologies.

**Future Enhancements:**

* Real-time surveillance with live camera feeds.
* Integration with IoT devices for automated alerts and monitoring.
* Augmented Reality (AR) navigation for immersive guidance.

**Objective:**

The objective of this project is to develop a secure and tamper-proof evidence management system that utilizes blockchain technology to store, manage, and track evidence for forensic, legal, and law enforcement purposes.

**Abstract:**

Traditional evidence management systems are prone to tampering, data loss, and inefficiencies in tracking the chain of custody. The **Blockchain-Based Evidence Management System** leverages the immutable and transparent nature of blockchain technology to address these challenges. The system ensures the integrity and security of digital evidence while maintaining a verifiable audit trail. Role-based access is provided for administrators, forensic staff, and police, ensuring data privacy and operational efficiency.

**Features:**

1. **Secure Evidence Storage:**
   * Stores metadata of evidence (e.g., timestamps, case details) on the blockchain for immutability.
   * Links the actual evidence (e.g., images, videos, documents) stored in IPFS (InterPlanetary File System).
2. **Role-Based Access Control:**
   * **Admin:** Manages users, assigns roles, and oversees system operations.
   * **Forensic Staff:** Uploads evidence to the blockchain and IPFS.
   * **Police:** Views and verifies evidence for investigations.
3. **Chain of Custody:**
   * Maintains an audit trail of evidence handling, ensuring transparency and accountability.
4. **Tamper-Proof Records:**
   * Blockchain's immutability ensures that no unauthorized modifications can occur.
5. **Decentralized Architecture:**
   * Ensures reliability and prevents single points of failure by utilizing blockchain and IPFS.

**Technology Stack:**

* **Frontend:** React.js, HTML, CSS, JavaScript, Bootstrap
* **Backend:** Node.js, Express.js
* **Blockchain Framework:** Ethereum (using Solidity and Truffle framework)
* **Storage:** IPFS for digital evidence files
* **Database:** MySQL (for user data and additional metadata)
* **Development Tools:** Visual Studio Code, Ganache, and MetaMask

**Implementation Details:**

1. **Blockchain Smart Contracts:**
   * Define smart contracts using Solidity for:
     + Evidence creation and storage.
     + Role-based access.
     + Chain of custody tracking.
   * Deploy smart contracts using Truffle and Ganache.
2. **Decentralized File Storage:**
   * Store digital evidence (e.g., images, videos) in IPFS.
   * Use the generated IPFS hash to reference files in the blockchain.
3. **Frontend Design:**
   * User-friendly web interface using React.js.
   * Role-specific dashboards for admins, forensic staff, and police.
4. **Backend APIs:**
   * Use Node.js and Express.js to handle API requests and communicate with the blockchain.
5. **Role-Based Access Control:**
   * Implement JWT (JSON Web Tokens) for user authentication and authorization.

**Step-by-Step Procedure:**

1. **Set Up Development Environment:**
   * Install Node.js, Truffle, and Ganache.
   * Set up IPFS for decentralized file storage.
2. **Create Smart Contracts:**
   * Define EvidenceManagement.sol for evidence storage and access.
   * Compile and deploy the contract using Truffle.
3. **Backend Development:**
   * Set up Node.js server with APIs to interact with smart contracts and IPFS.
   * Store user roles and metadata in MySQL.
4. **Frontend Development:**
   * Build role-specific dashboards using React.js.
   * Integrate MetaMask for blockchain interactions.
5. **Testing:**
   * Test smart contracts on the Ganache blockchain.
   * Verify the IPFS integration for file storage and retrieval.
6. **Deployment:**
   * Deploy the system on Ethereum Testnet (e.g., Ropsten) or private blockchain.
   * Host the frontend and backend on platforms like Vercel, Heroku, or AWS.

**Benefits:**

1. **Data Integrity:** Ensures evidence cannot be altered or tampered with.
2. **Transparency:** Provides a verifiable audit trail for all evidence-related transactions.
3. **Security:** Decentralized architecture minimizes the risk of data breaches.
4. **Efficiency:** Simplifies evidence tracking and access through automation.

**Applications:**

* Law enforcement agencies to manage criminal case evidence.
* Legal systems to ensure reliable evidence submission in courts.
* Forensic laboratories for secure evidence storage and tracking.

**Future Enhancements:**

* Integration of AI for automatic evidence categorization and analysis.
* Mobile app support for uploading and accessing evidence on the go.
* QR code-based evidence tagging for physical items.